User code
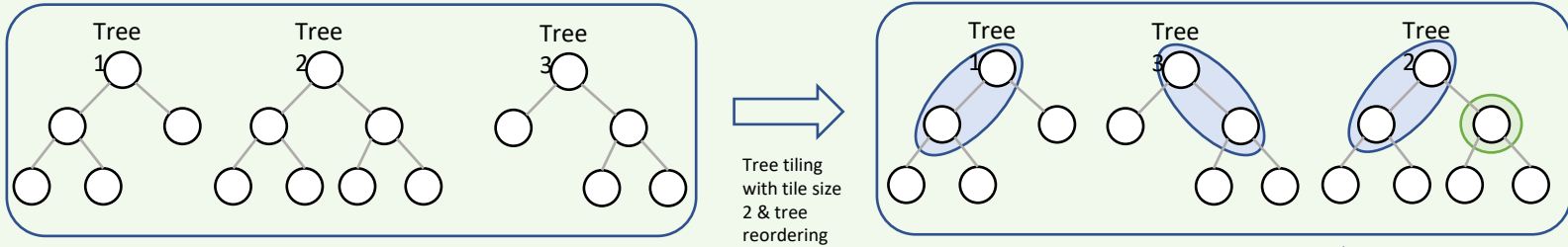
```
inferenceRunner = treebeard.FromModelFile(modelFile, options)
results = inferenceRunner.RunInference(batch)
```

High level IR

Tree 1   Tree 2   Tree 3

Tree tiling with tile size 2 & tree reordering

Tree 1   Tree 3   Tree 2

Mid level IR

One tree at a time

One row at a time

```
for t = 0 to 1 step 1 {
  for i = 0 to batchSize step 1 {
    tree = getTree(forest, t)
    node = getRoot(tree)
    node = traverseTile(node, rows[i])
    treePrediction = getLeafValue(node)
    prediction[i] = prediction[i] + treePrediction
  }
}
for t = 1 to 3 step 1 {
  for i = 0 to batchSize step 1 {
    tree = getTree(forest, t)
    node = getRoot(tree)
    node = traverseTile(node, rows[i])
    node = traverseTile(node, rows[i])
    treePrediction = getLeafValue(node)
    prediction[i] = prediction[i] + treePrediction
  }
}
```

Unroll tree walks

```
for t = 0 to 1 step 1 {
  for i = 0 to batchSize step 1 {
    tree = getTree(forest, t)
    treePrediction = WalkDecisionTree(tree, rows[i])
    prediction[i] = prediction[i] + treePrediction
  }
}
for t = 1 to 3 step 1 {
  for i = 0 to batchSize step 1 {
    tree = getTree(forest, t)
    treePrediction = WalkDecisionTree(tree, rows[i])
    prediction[i] = prediction[i] + treePrediction
  }
}
```

```
for i = 0 to batchSize step 1 {
  prediction = 0
  for t = 0 to 1 step 1 {
    tree = getTree(forest, t)
    treePrediction = WalkDecisionTree(tree, rows[i])
    prediction = prediction + treePrediction
  }
  for t = 1 to 3 step 1 {
    tree = getTree(forest, t)
    treePrediction = WalkDecisionTree(tree, rows[i])
    prediction = prediction + treePrediction
  }
  predictions[i] = prediction
}
```

Low level IR

Vectorize

```
// …
// Low level IR for single traverseTile
thresholds = loadThresholds(tree, n)
featureIndices = loadFeatureIndices(tree, n)
// Gather required feature from the current row
features = rows[i][featureIndices]
// Vector comparison of features and thresholds
comparison = features < thresholds

// Pack bits in comparison vector into an integer
comparisonIndex = combineBitsIntoInt(comparison)

// Get child index of tile we need to move to
tileShape = loadTileShape(tree, n)
childIndex = LUT[tileShapeID, comparisonIndex]

// Move to the correct child of the current node
n = getChildNode(tree, n, childIndex)
// …
```

Lower to LLVM IR